Dimensionality Reduction

전 재 욱

Embedded System 연구실
성균관대학교

Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Outline

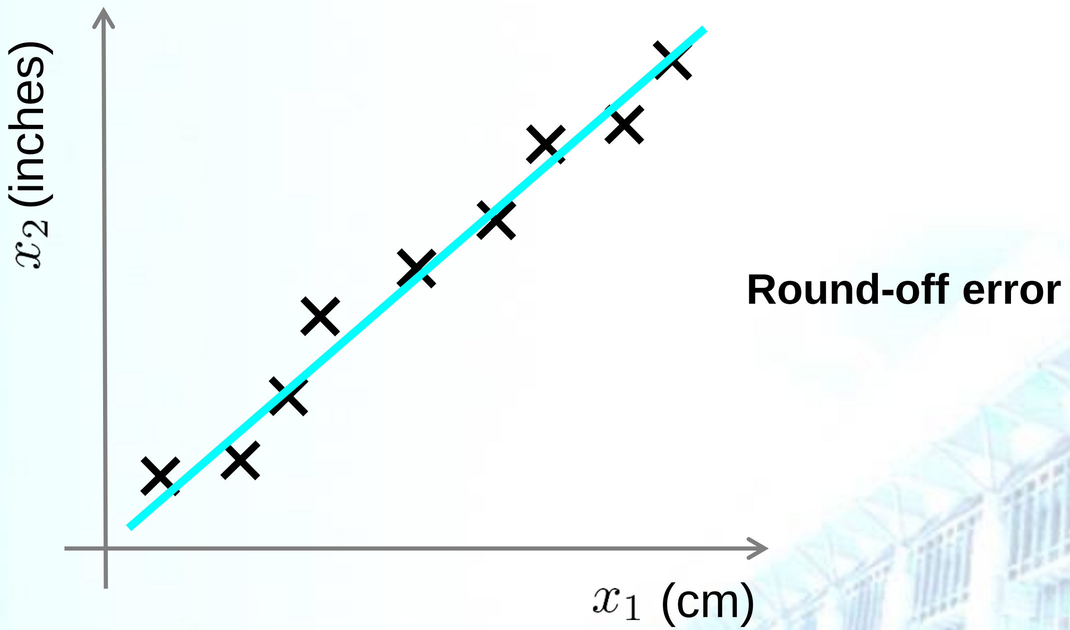
- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Data Compression

- Dimensionality reduction
  - One type of unsupervised learning problem

Data Compression

- What is dimensionality reduction?
  - More collected features than needed
    - Can we "simplify" our data set in a rational and useful way?
  - Example
    - Redundant data set - different units for same attribute
      - Reduce data to 1D (2D ➔ 1D)

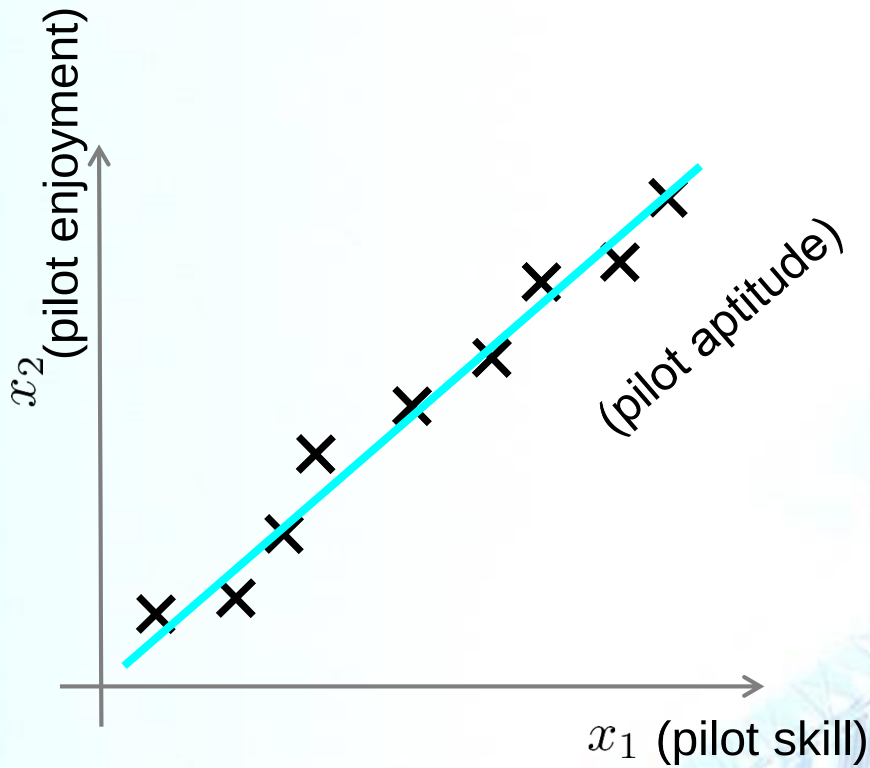


Data Compression

- Data redundancy can happen when different teams are working independently
  - Often generates redundant data (especially if we do not control data collection)

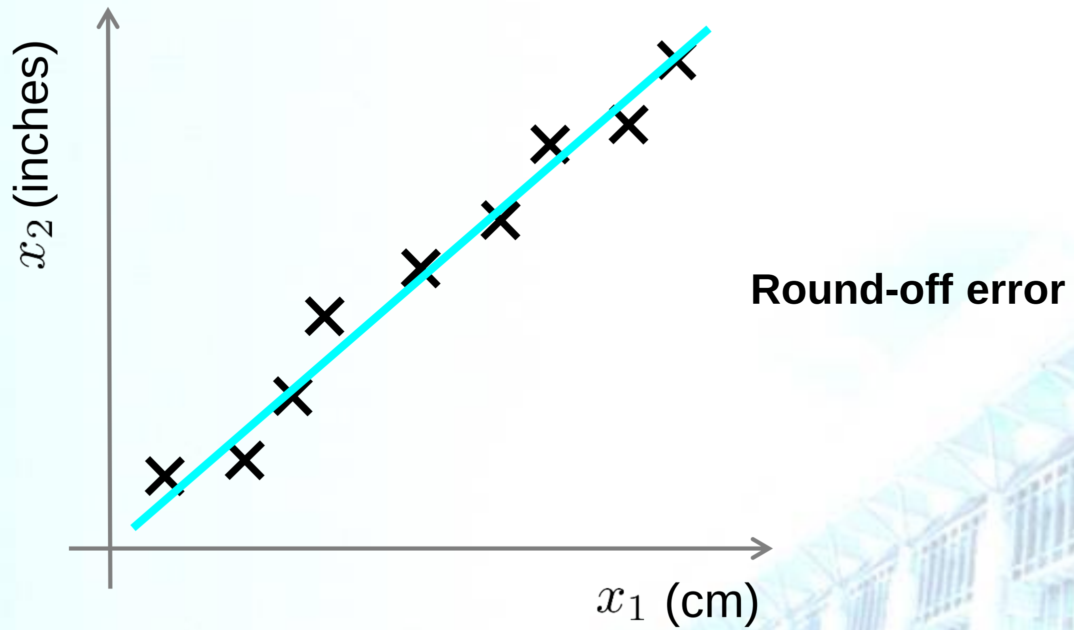
Data Compression

- Another example
  - Helicopter flying
    - A survey of pilots (x_1: skill, x_2: pilot enjoyment)
      - These features may be highly correlated
      - This correlation can be combined into a single attribute called aptitude (for example)

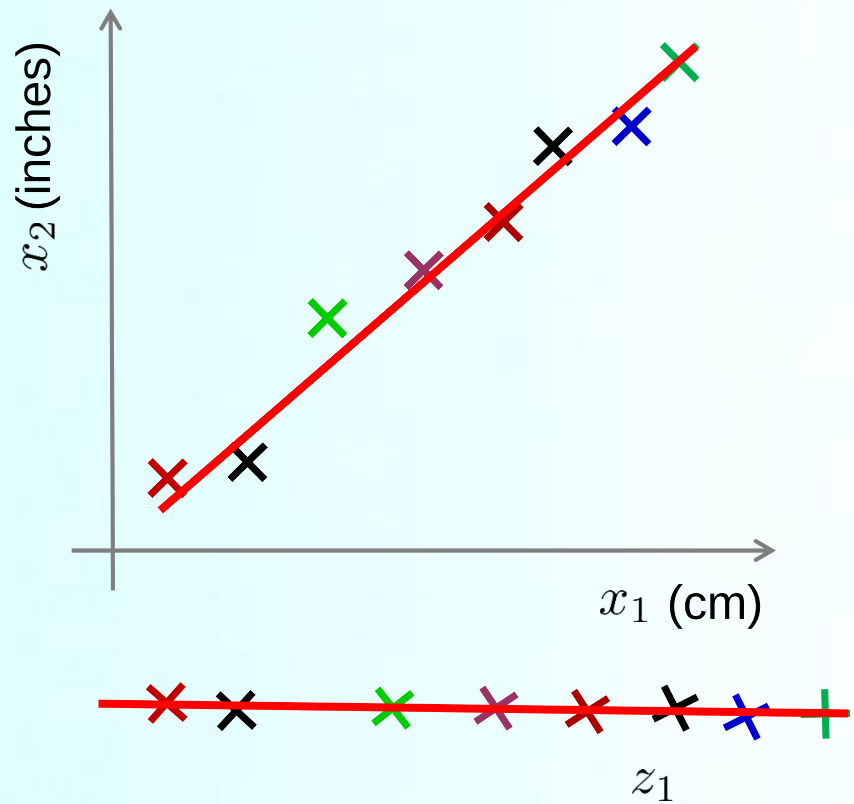


Data Compression

- What is dimensionality reduction?
  - More collected features than needed
    - Can we "simplify" our data set in a rational and useful way?
  - Example
    - Redundant data set - different units for same attribute
      - Reduce data to 1D (2D ➔ 1D)



Data Compression



- Reduce data from 2D to 1D

  $x^{(1)} \in R^2 \rightarrow z^{(1)} \in R$

  $x^{(2)} \in R^2 \rightarrow z^{(2)} \in R$

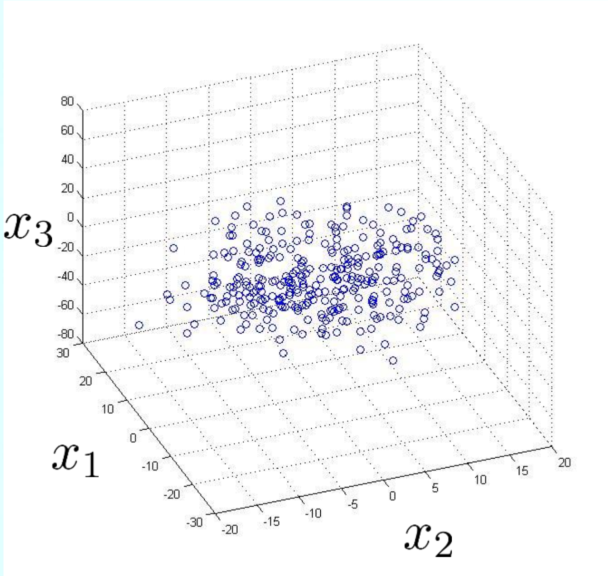
  $\vdots$

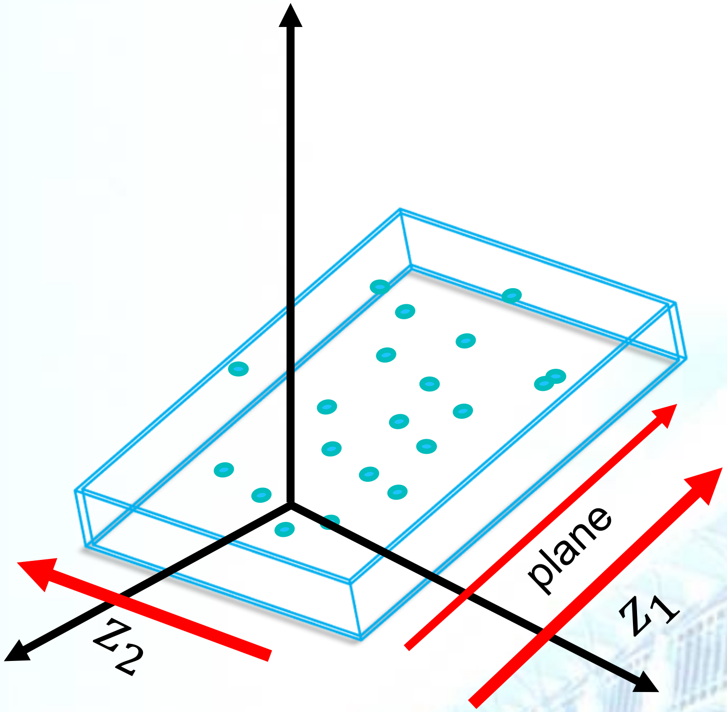
  $x^{(m)} \in R^2 \rightarrow z^{(m)} \in R$

  - So we can approximate original examples
    - Allows us to half the amount of storage

Data Compression

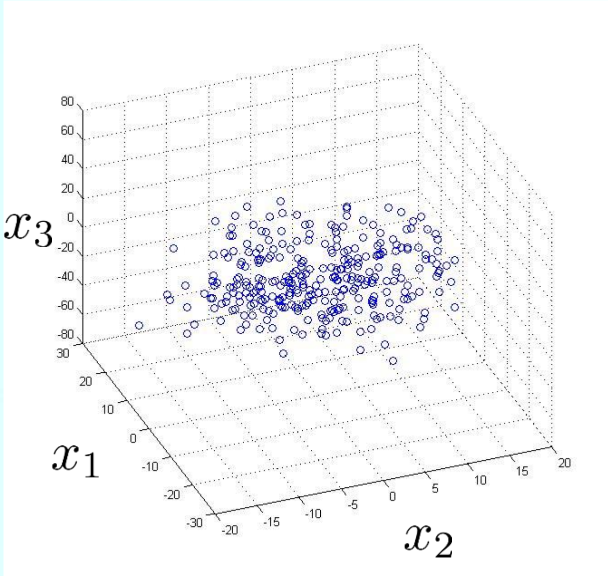
- Reduce data from 3D to 2D
  - The following data are given: $x^{(i)} \in R^3$
    - All the data may lie in one plane
      - e.g. all are sitting inside the right shallow tray
        - So, one of dimension can be ignored.
        - Can draw two new axis along the plane of this tray

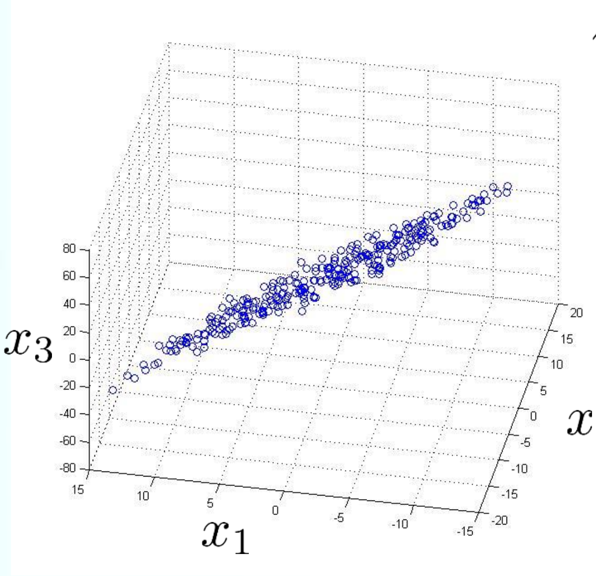


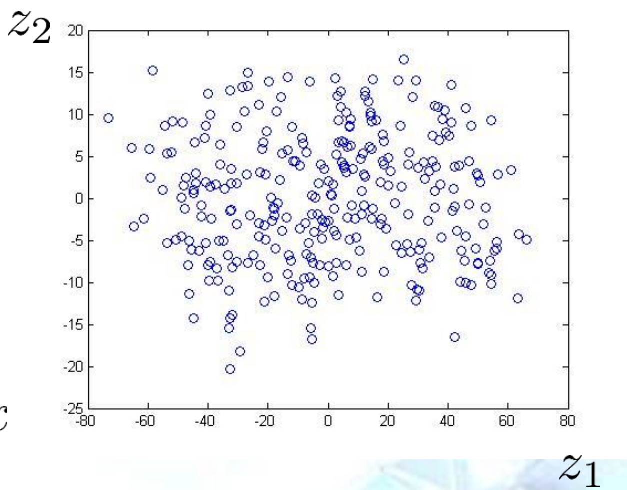


Data Compression

- Reduce data from 3D to 2D
  - $x^{(i)} \in R^3,\ z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix} \in R^2$

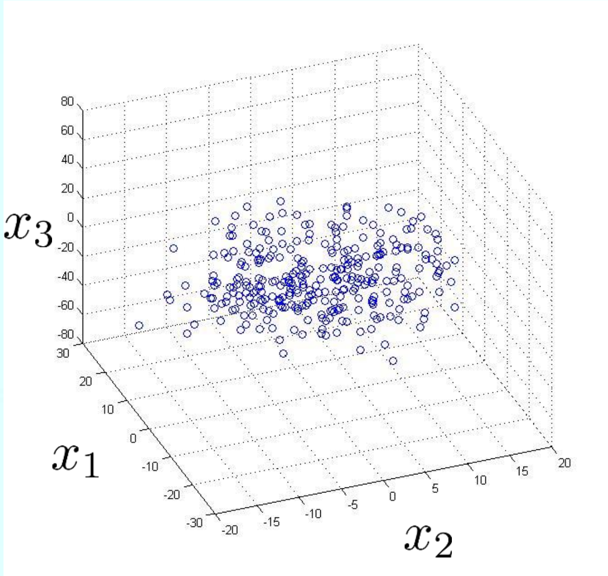


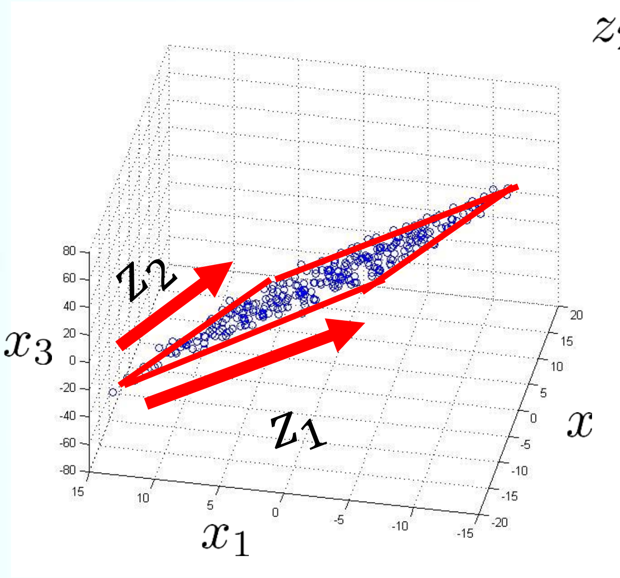


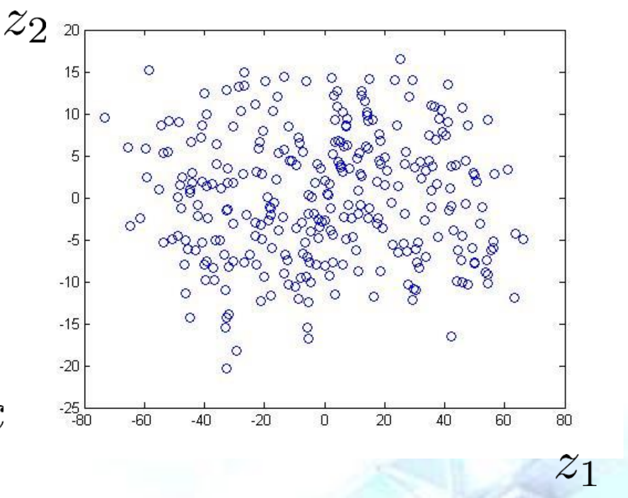


Data Compression

- Reduce data from 3D to 2D
  - $x^{(i)} \in R^3,\ z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix} \in R^2$







  - In reality, we would normally try and do 1,000D ➔ 100D

Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Visualization

- Hard to visualize highly dimensional data
  - Dimensionality reduction can improve how we display information in a tractable manner for human
  - Why do we care?
    - Often helps to develop algorithms if we can understand our data better
      - Dimensionality reduction helps us do this
    - Good for explaining something to someone if we can "show" it in the data

Data Visualization

- Example
  - A large data set about many facts of a country around the world

Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Development Index	Life Expectancy	Poverty Index (Gini as percentage)	Mean household Income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

  - x_1: GDP, … , x_6: Mean household income
  - Assume 50 features per country
    - How can we understand this data better?
      - Very hard to plot 50 dimensional data

Data Visualization

- Using dimensionality reduction,
  - instead of each country being represented by a 50D feature vector
    - Come up with a different feature representation (z values) which summarize these features

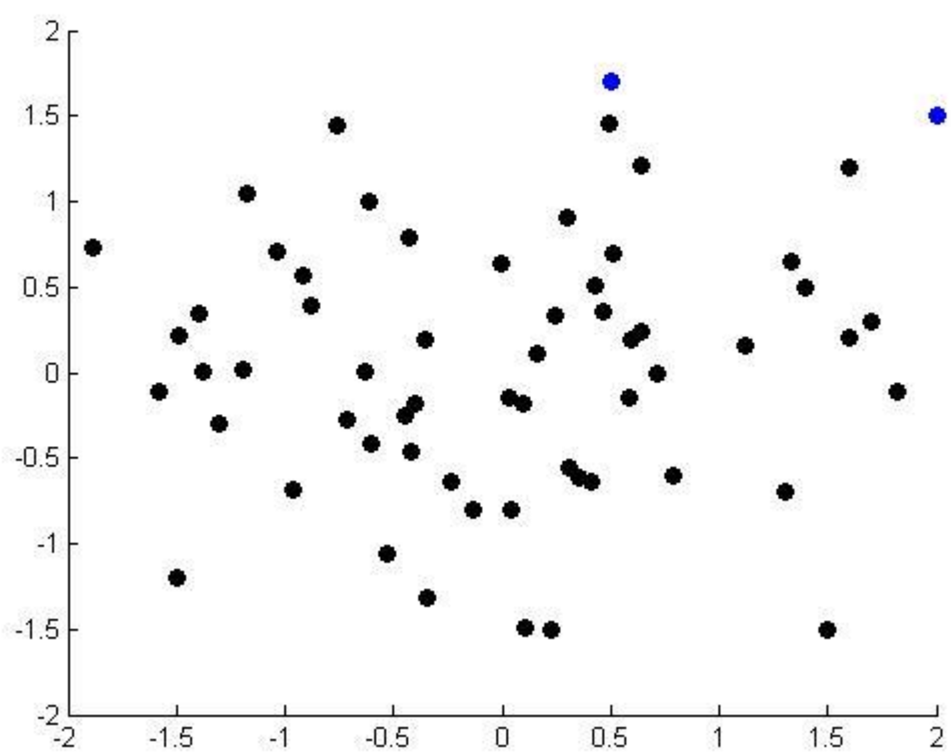
Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

  - This gives us a 2-dimensional vector
    - Reduce 50D ➔ 2D
    - Plot as a 2D plot

Data Visualization

Per person GDP
(economic activity)

z_2



z_1

Country size (GDP)

Data Visualization

- Typically we do not generally ascribe meaning to the new features
  - so we have to determine what these summary values mean
- So despite having 50 features, there may be two "dimensions" of information, with features associated with each of those dimensions
  - It is up to us to asses what of the features can be grouped to form summary features, and how best to do that (feature scaling is probably important)
- Helps show the two main dimensions of variation in a way that is easy to understand

Outline

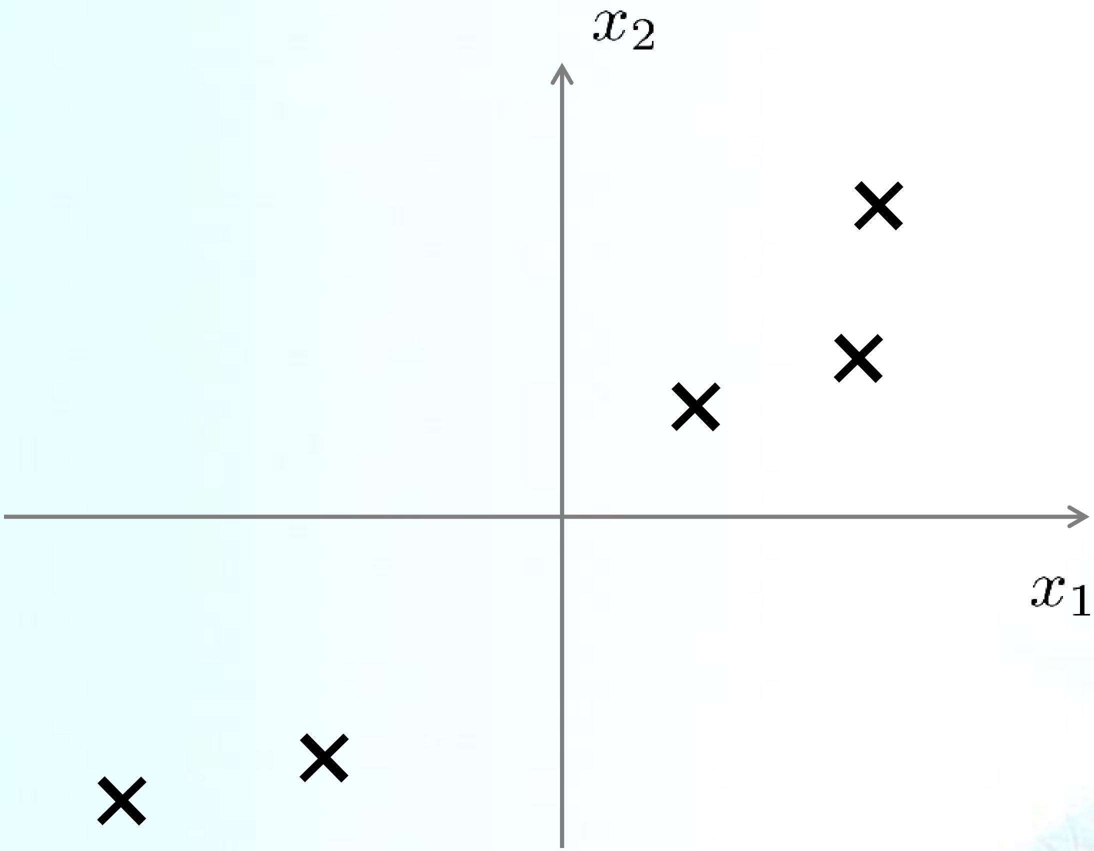
- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

PCA Problem Formulation

- PCA(Principle Component Analysis)
  - The most commonly used algorithm
    for the problem of dimensionality reduction

PCA Problem Formulation

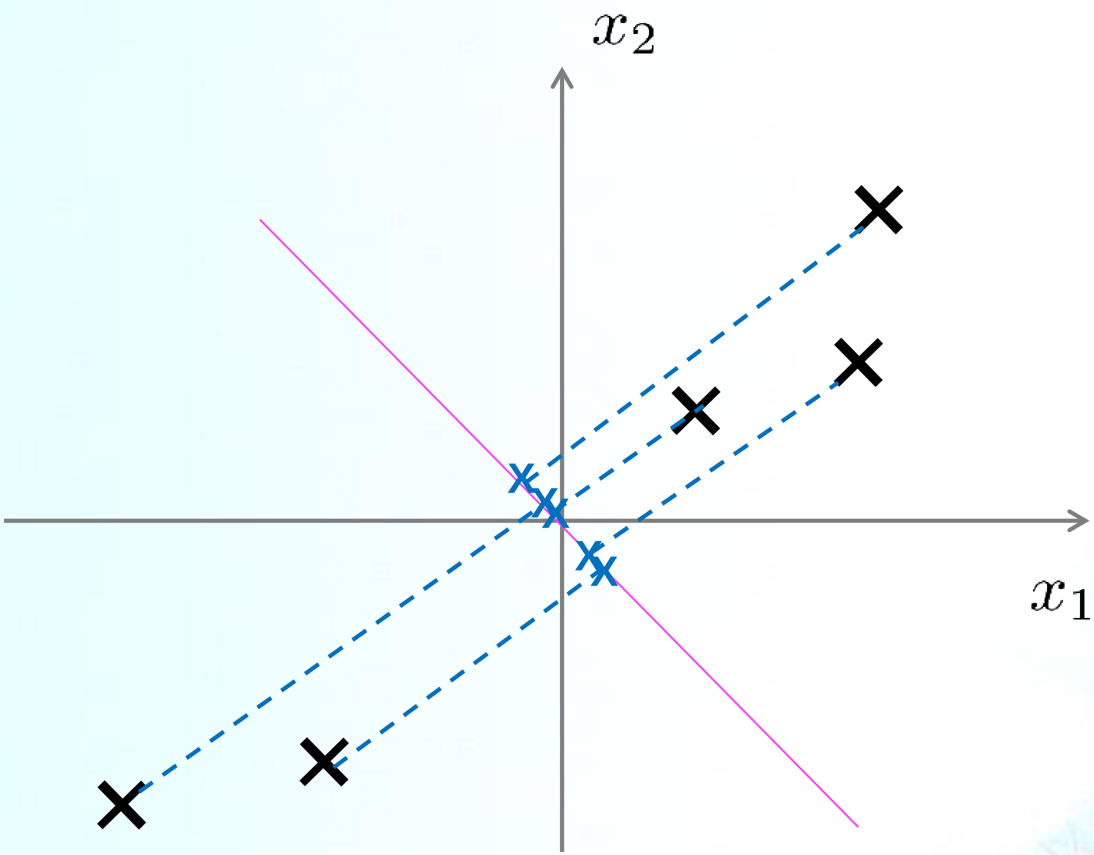
- PCA(Principle Component Analysis)
  - For example, we have a 2D data set which we wish to reduce to 1D
    - In other words, find a single line onto which to project this data



Embedded System Lab.

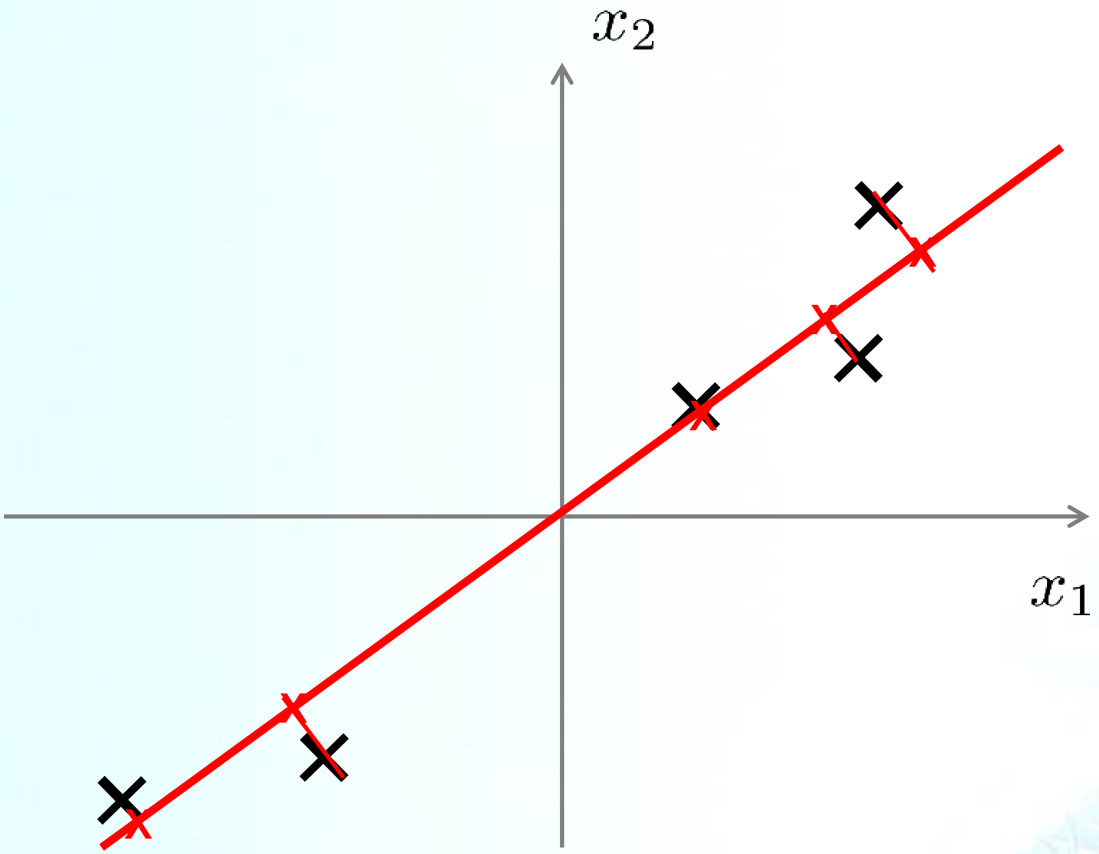
PCA Problem Formulation

- PCA(Principle Component Analysis)



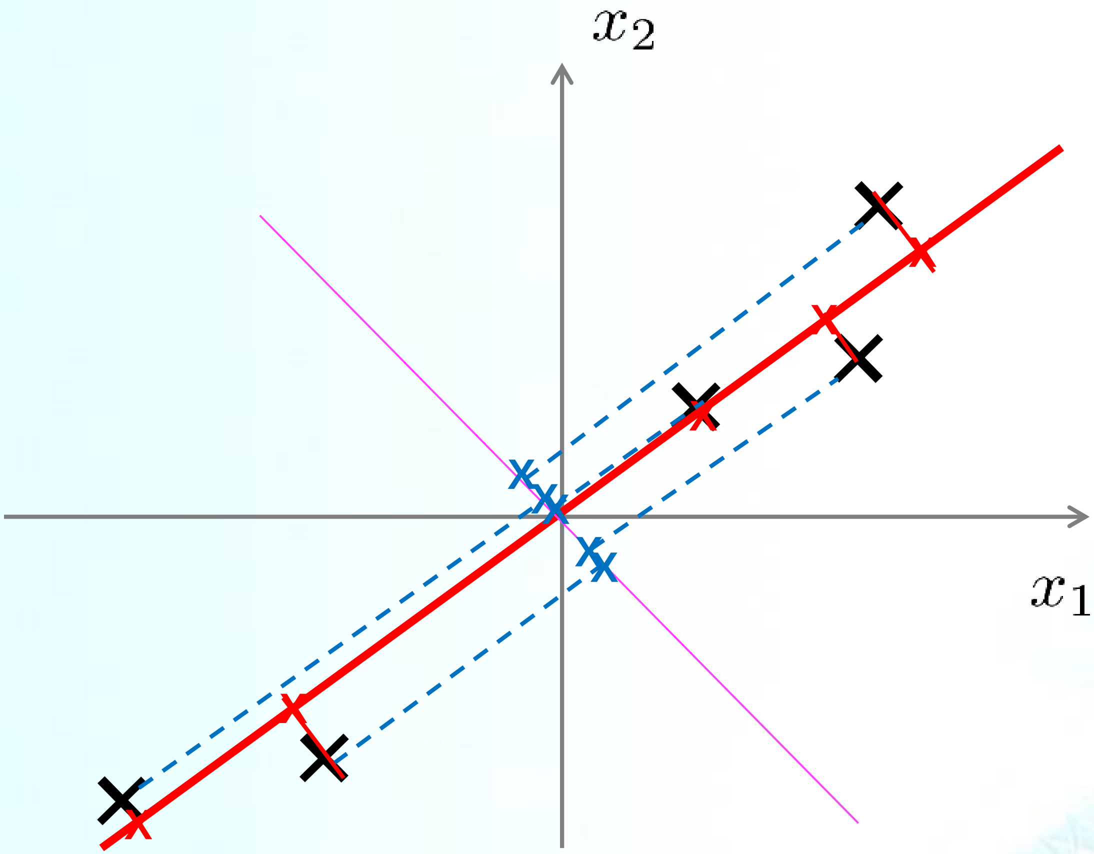
PCA Problem Formulation

- PCA(Principle Component Analysis)



PCA Problem Formulation

- PCA(Principle Component Analysis)



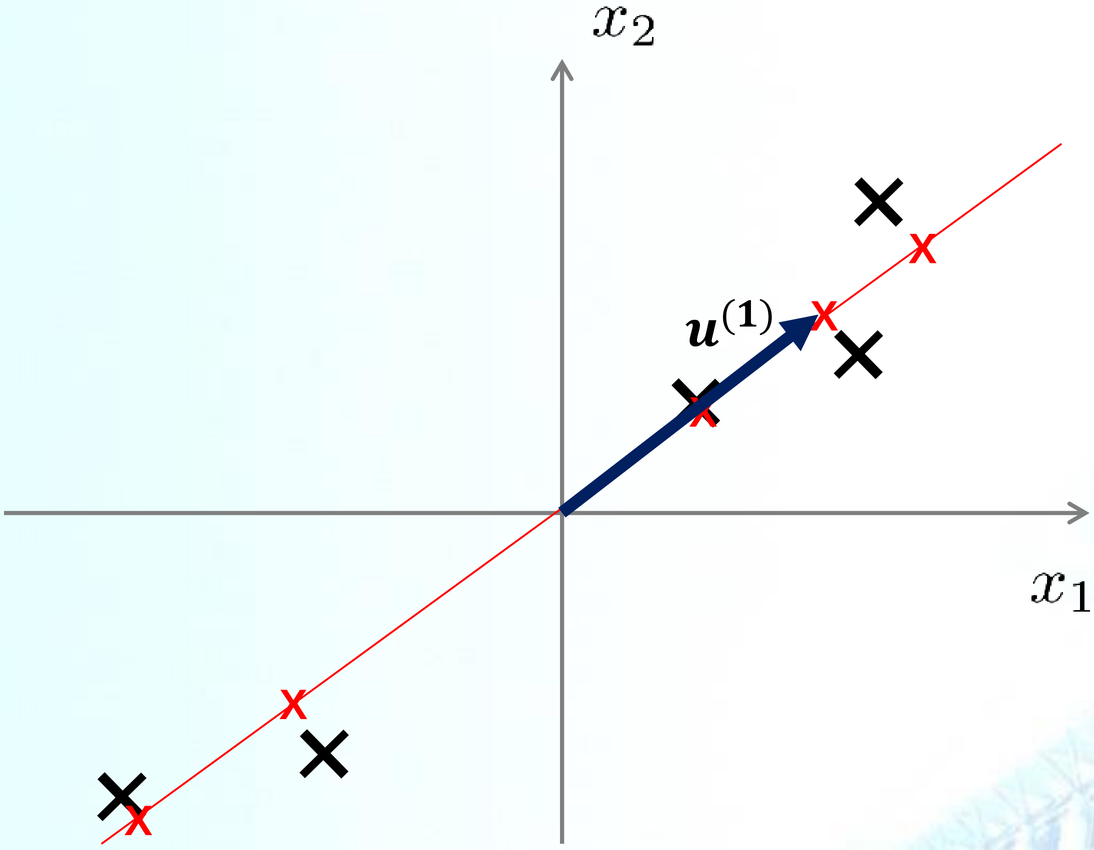
PCA Problem Formulation

- PCA(Principle Component Analysis)
  - Distance btw each point and the projected version should be small
    - (red lines in the previous slide are short)
  - PCA tries to find a lower dimensional surface

    so the sum of squares onto that surface is minimized
  - The red lines are sometimes called the **projection error**
    - PCA tries to find the surface (a straight bold red line in the previous slide) which has the minimum projection error
  - Before applying PCA,
    - we should normally do **mean normalization** and **feature scaling** on our data

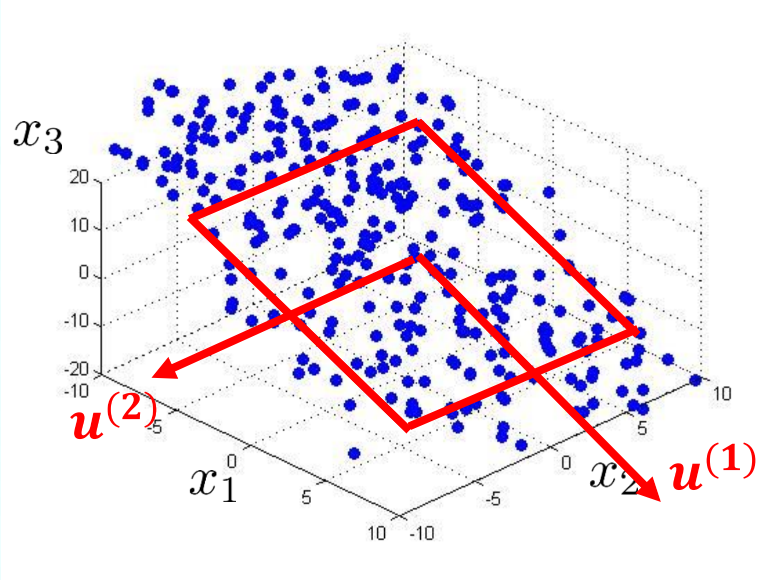
PCA Problem Formulation

- PCA(Principle Component Analysis)
  - Reduce from 2-dim to 1-dim:
    - Find a direction (a vector $u^{(1)} \in R$) onto which to project the data so as to minimize the projection error.



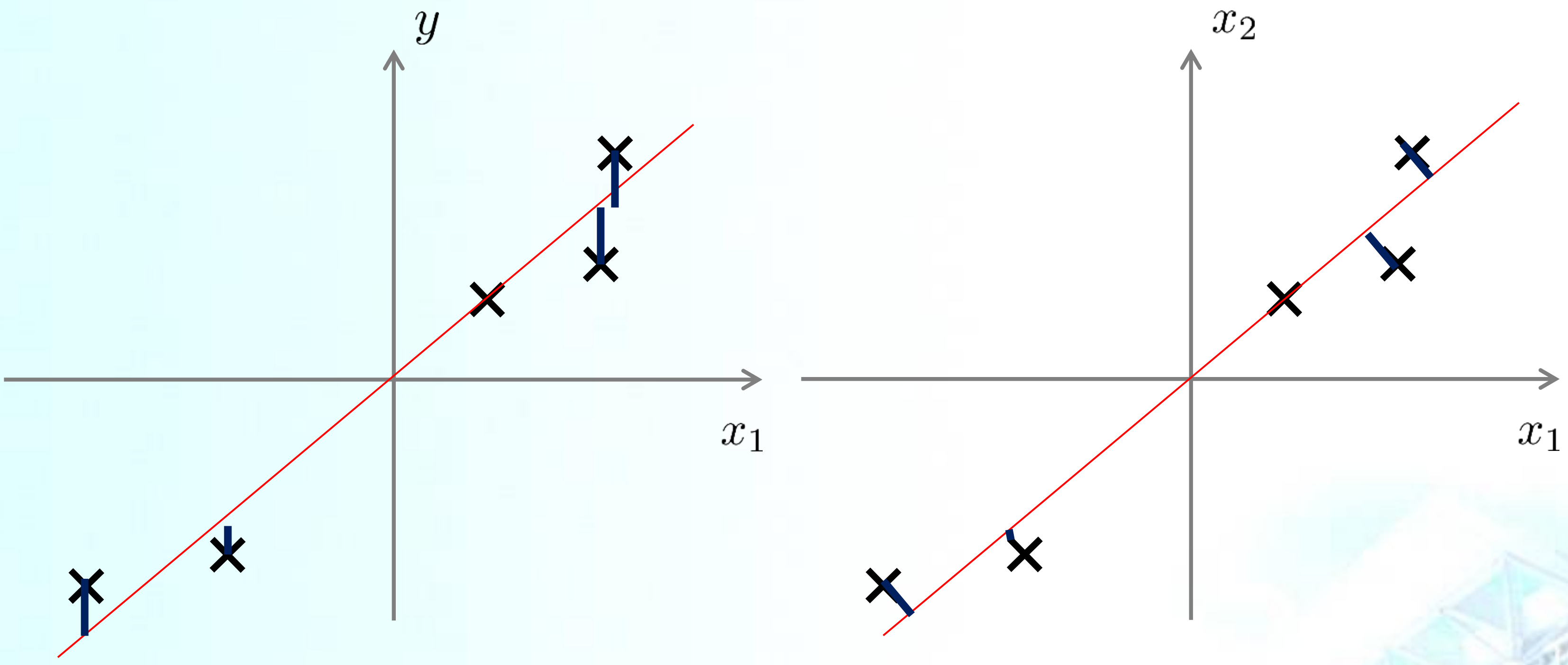
PCA Problem Formulation

- PCA(Principle Component Analysis)
  - Reduce from n-dim to k-dim:
    - Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.
    - (e.g. 3D ➔ 2D)



PCA Problem Formulation

- PCA is **not** linear regression



PCA Problem Formulation

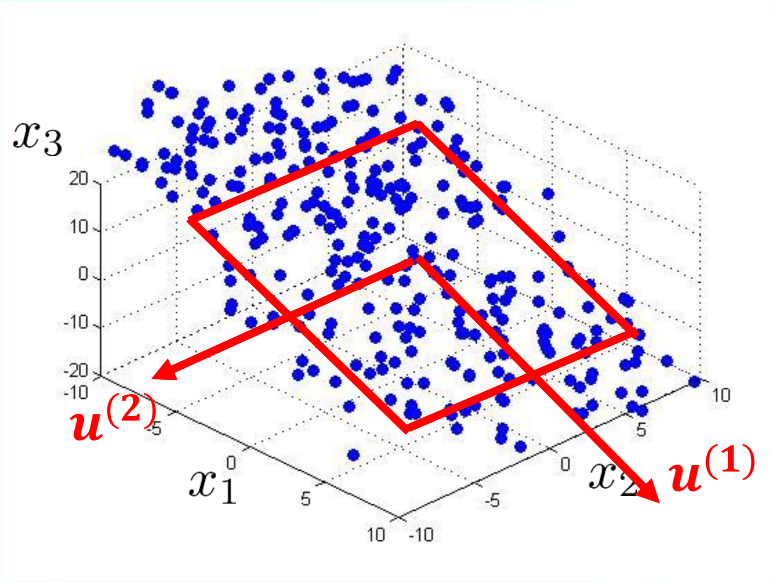
- PCA is **not** linear regression
  - For linear regression,
    - fitting a straight line to minimize the **straight line** btw a point $\left(x^{(i)}, y^{(i)}\right)$ and its predicted point $\left(x^{(i)}, ax^{(i)} + b\right)$
      - **VERTICAL distance** btw points
  - For PCA,
    - minimizing the magnitude of the shortest **orthogonal distance**
    - Gives very different effects

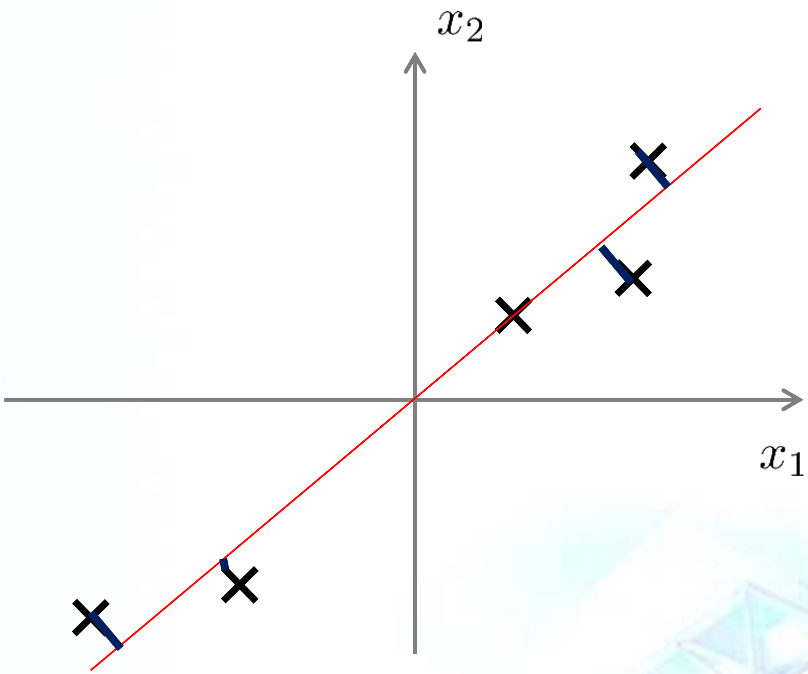
PCA Problem Formulation

- PCA is **not** linear regression
  - More generally
    - With linear regression
      - We are trying to predict "y"
    - With PCA
      - there is no "y“
        - Instead we have a list of features and all features are treated equally
      - If we have 3D dimensional data 3D ➔2D
        - Have 3 features treated symmetrically

PCA Problem Formulation

- PCA is **not** linear regression





Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

PCA Algorithm

- Given a training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

  data preprocessing is necessary before applying PCA

  - Mean normalization

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$

  - Feature scaling (depending on data)
    - If different features on different scales (e.g., x_1= size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

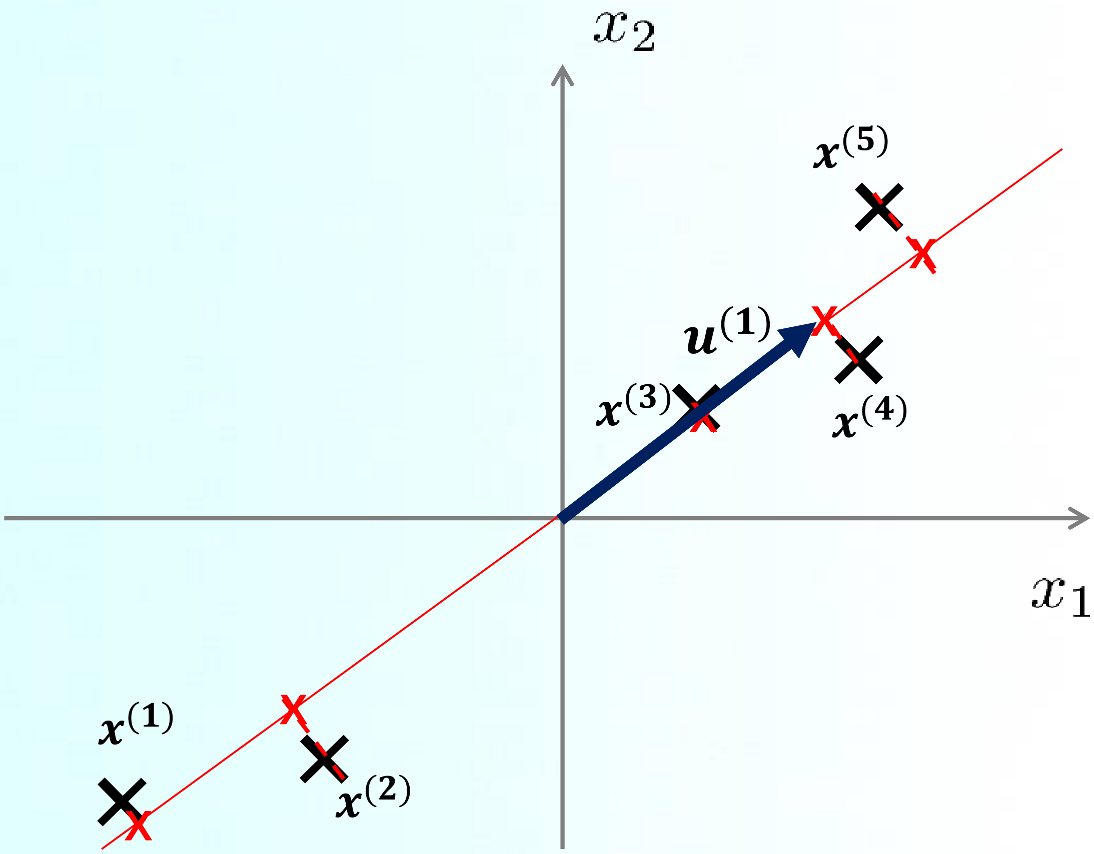
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

where s_j is some measure of the range

- biggest – smallest
- standard deviation (more commonly)

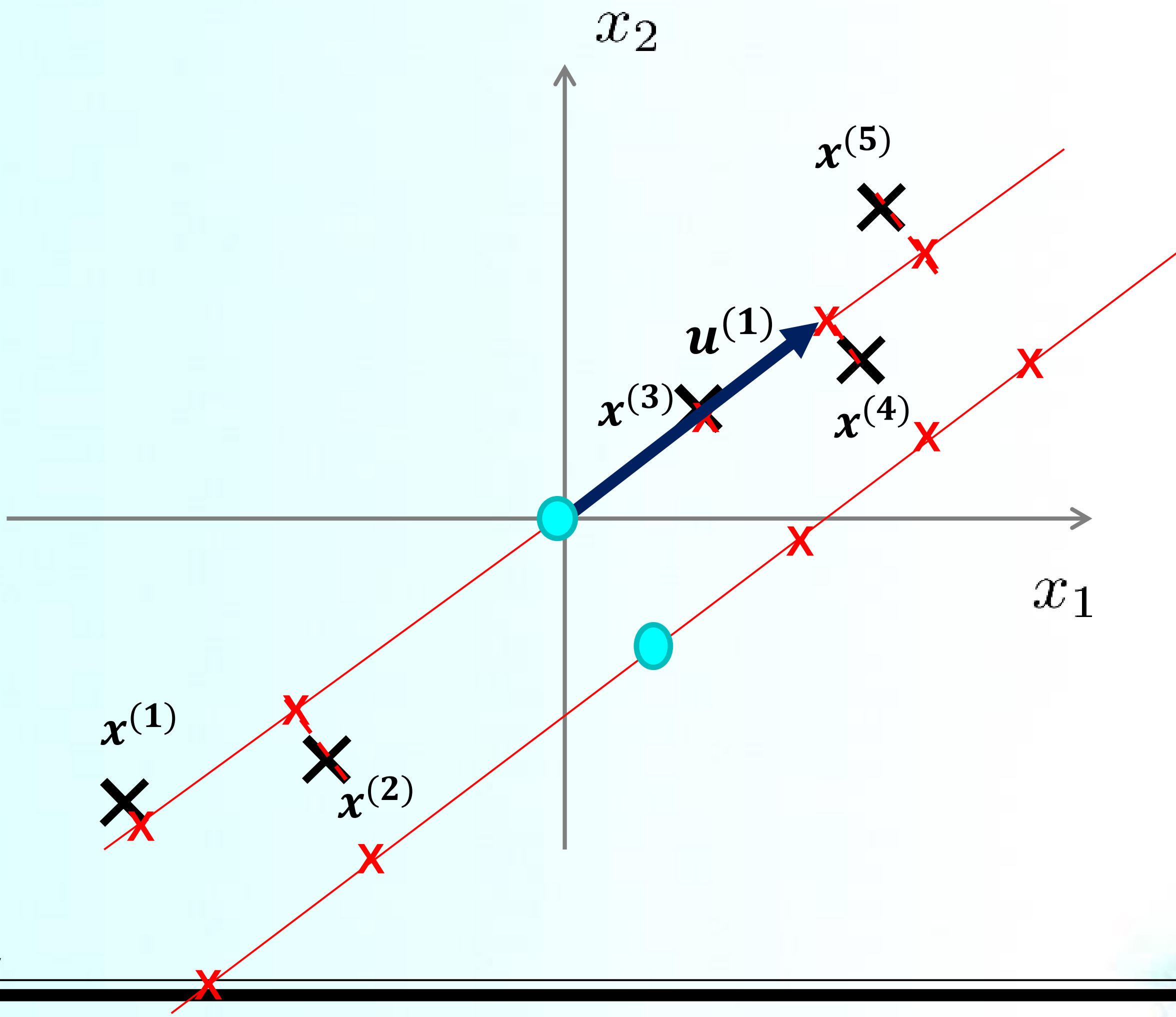
PCA Algorithm

- With preprocessing done,
  - PCA finds the lower dimensional sub-space which minimizes the sum of the square (e.g. 2D ➔ 1D, 3D ➔ 2D)



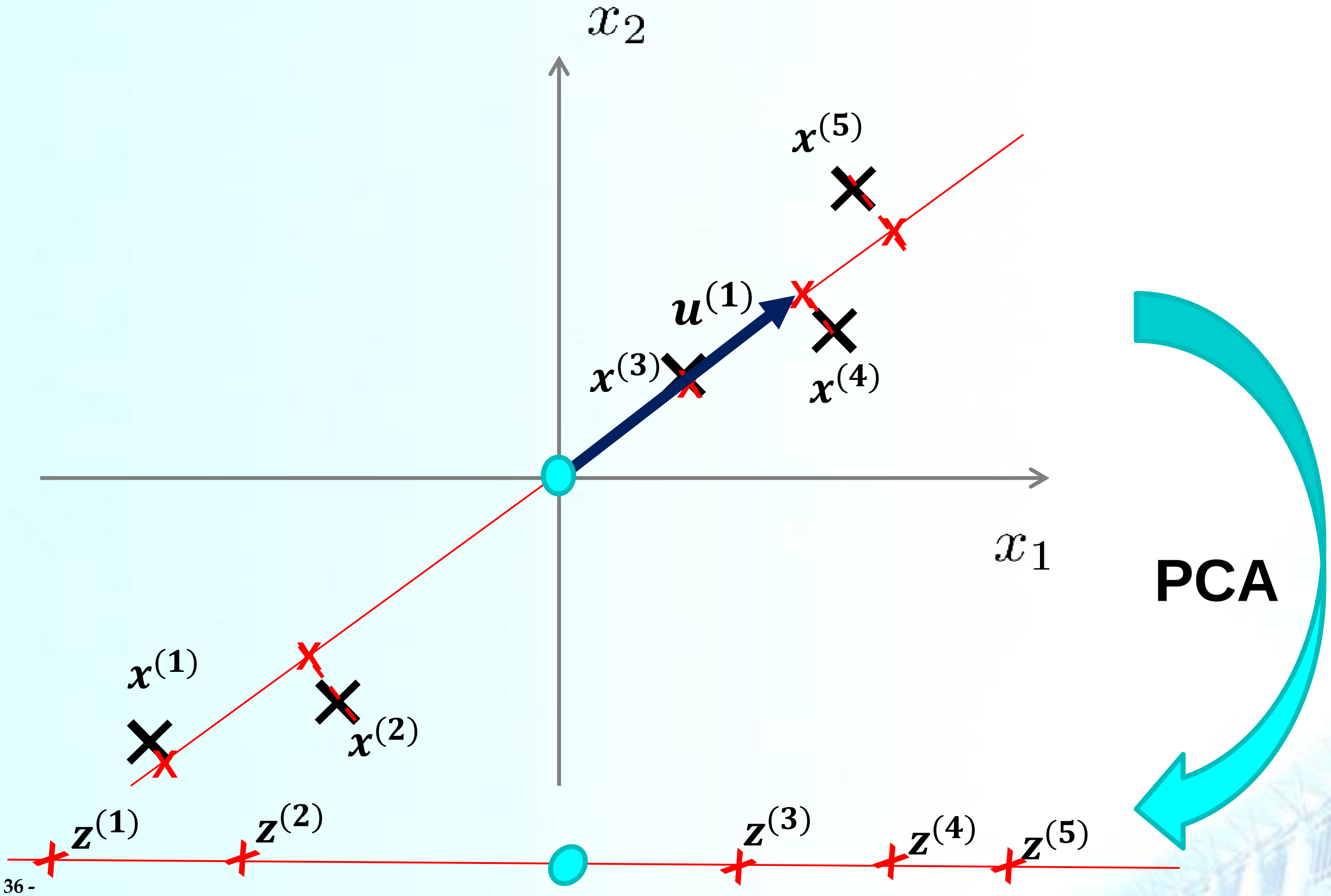
PCA Algorithm

- With preprocessing done,
  - PCA finds the lower dimensional sub-space which minimizes the sum of the square (e.g. 2D ➔ 1D, 3D ➔ 2D)



PCA Algorithm

- With preprocessing done,
  - PCA finds the lower dimensional sub-space which minimizes the sum of the square (e.g. 2D ➔ 1D, 3D ➔ 2D)

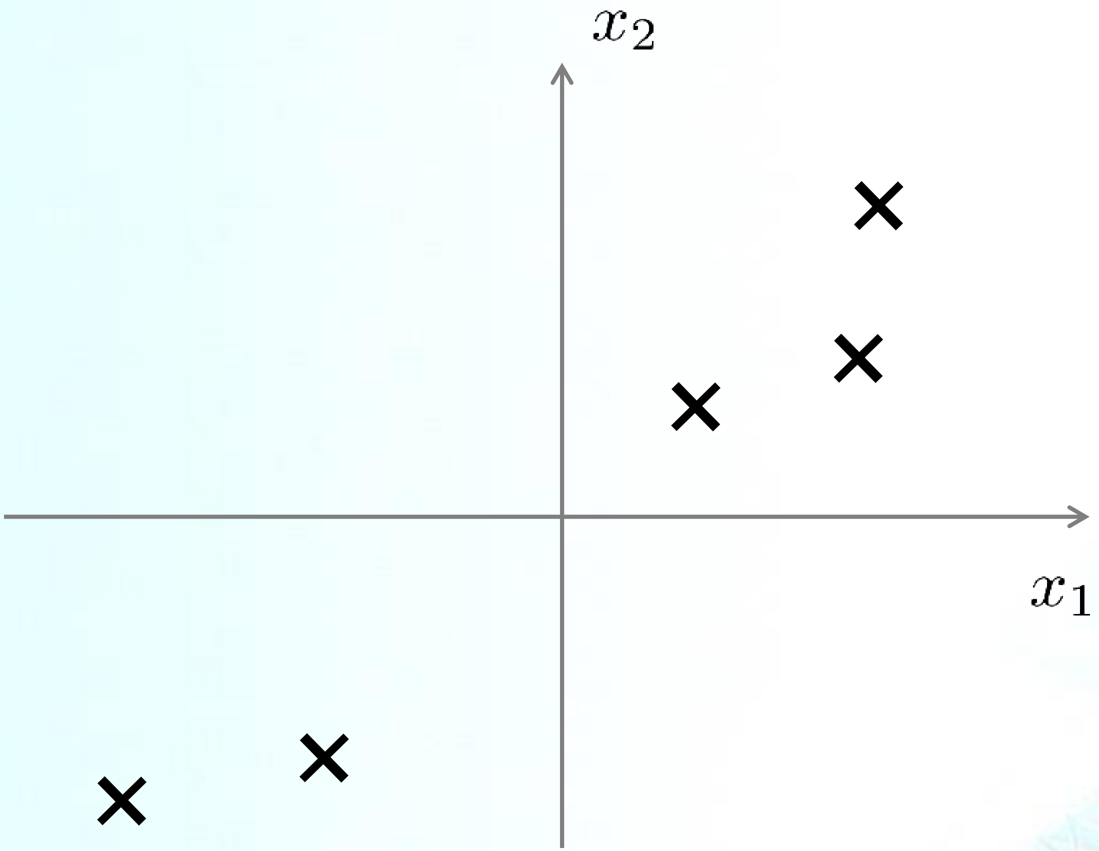


PCA Algorithm

- PCA need to compute two things;
  - Compute the **u vectors**
    - The new planes
  - Compute the **z vectors**
    - z vectors are the new, lower dimensionality feature vectors

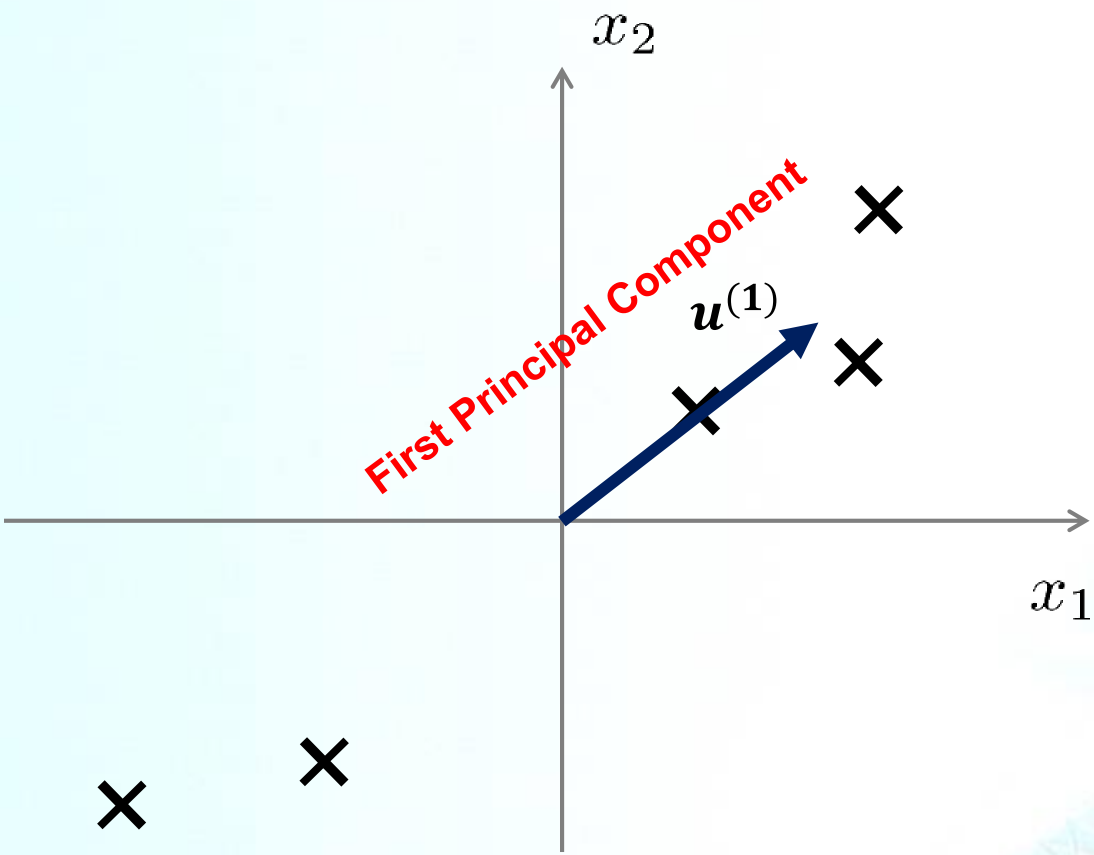
PCA Problem

- Reduce data from 2D to 1D



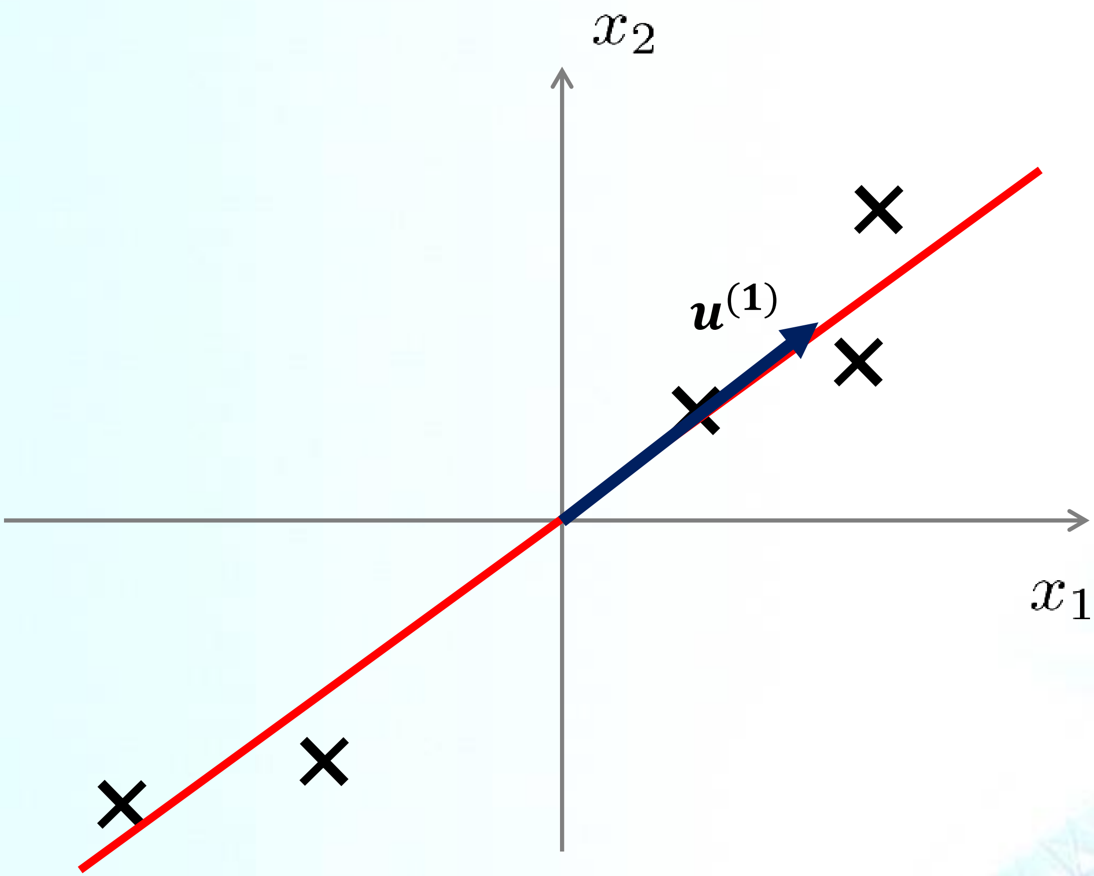
PCA Problem

- Reduce data from 2D to 1D



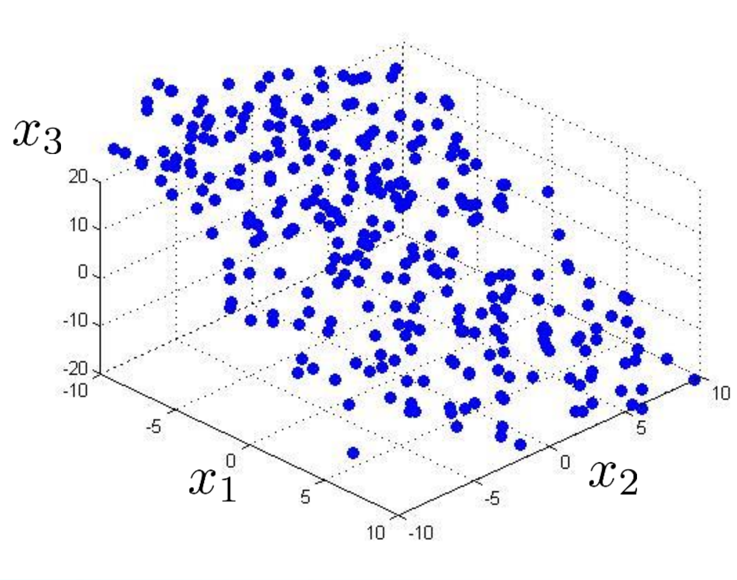
PCA Problem

- Reduce data from 2D to 1D



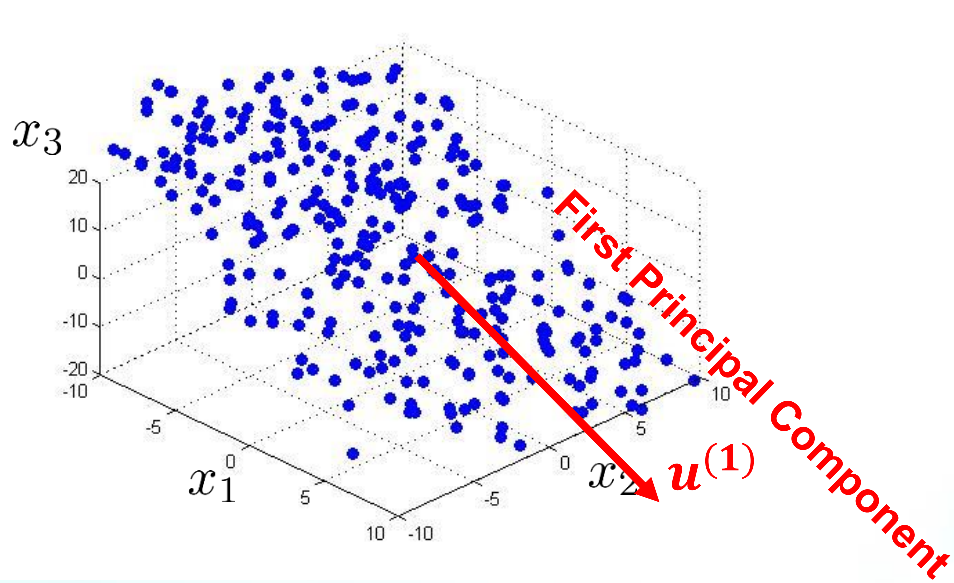
PCA Algorithm

- Reduce data from 3D to 2D.
  - $x^{(i)} \in R^3$ ➔ $z^{(i)} \in R^2$



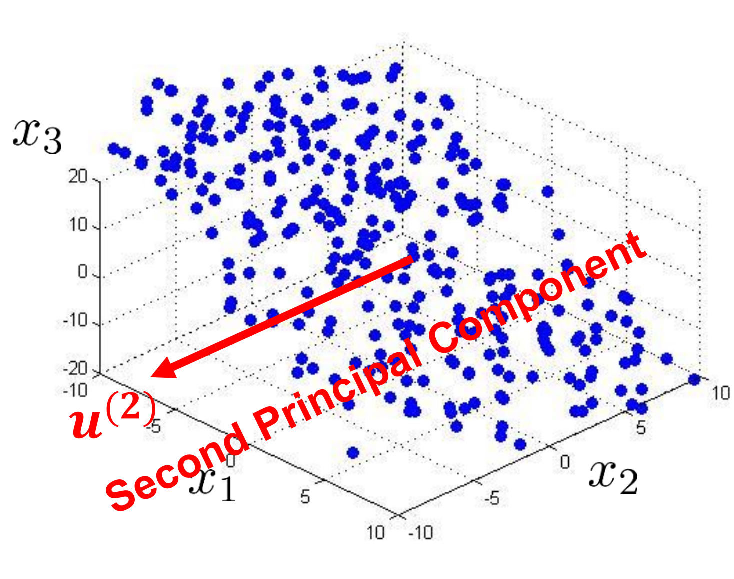
PCA Algorithm

- Reduce data from 3D to 2D.
  - $x^{(i)} \in R^3$ ➔ $z^{(i)} \in R^2$



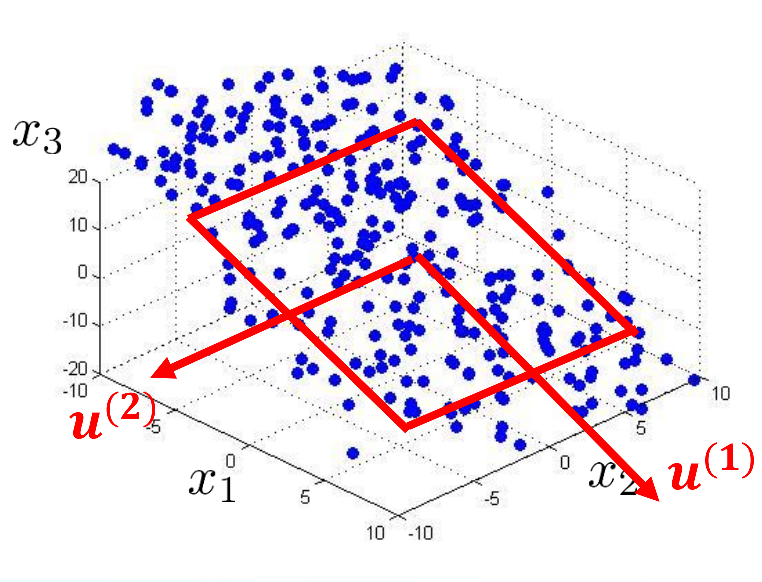
PCA Algorithm

- Reduce data from 3D to 2D.
  - $x^{(i)} \in R^3 \rightarrow z^{(i)} \in R^2$



PCA Algorithm

- Reduce data from 3D to 2D.
  - $x^{(i)} \in R^3$ ➔ $z^{(i)} \in R^2$



Singular Value Decomposition (SVD)

- Given an $m \times n$ matrix, A,

  then there exits a factorization

$$A = U\Sigma V^*$$

where

U is an $m \times m$ unitary matrix (i.e. $UU^* = U^*U = I$)

- If $U \in R^{m\times m}$, U is orthogonal s.t. $UU^T = U^TU = I$
- Column vectors form a set of orthonormal vectors
  - $u_i^T u_i = 1$ and $u_i^T u_j = 0$ for $i \neq j$

Σ is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal,

- The diagonal entries σ_i of Σ are known as the singular values of A
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ for $m \geq n$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$ for $m < n$

V is an $n \times n$ unitary matrix

Singular Value Decomposition (SVD)

- Example 1
  - Given a 2×3 matrix, $A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$,

$$A = U\Sigma V^* = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{18}} & -\frac{1}{\sqrt{18}} & \frac{4}{\sqrt{18}} \\ \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

Singular Value Decomposition (SVD)

- Example 2

  - Given a 3×3 matrix, $A = \begin{bmatrix} 0 & 1 & 1 \\ \sqrt{2} & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}$,

$$A = U\Sigma V^* = \begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} \\ \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{3}{\sqrt{12}} & \frac{1}{\sqrt{12}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

PCA Algorithm

- Reduce data from n – dim to k – dim
  - Compute "covariance matrix"

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}\boxed{\left(x^{(i)}\right)\left(x^{(i)}\right)^T} \quad \in \boldsymbol{R}^{n\times n}$$

  - Compute "eigenvectors" of matrix Σ
    - `[U,S,V] = svd(sigma)`
      - `svd` ➔ singular value decomposition
        - More numerically stable than **eig**
      - **eig** ➔ also gives eigenvector
  - U,S and V are matrices
    - U matrix: an [n x n] matrix
      - The columns of U are the **u** vectors we want
      - So to reduce a system from n – dim to k – dim
        - Just take the first *k-vectors* from U (first k columns)

$$\boldsymbol{U} = \begin{bmatrix} | & | & | & & | \\ \underbrace{\boldsymbol{u}^{(1)} & \boldsymbol{u}^{(2)} & \boldsymbol{u}^{(3)}}_{k} & \cdots & \boldsymbol{u}^{(n)} \\ | & | & | & & | \end{bmatrix}$$

$\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(k)}$

PCA Algorithm

- Reduce data from n –dim to k –dim
  - From `[U,S,V] = svd(sigma),`

$$\boldsymbol{U} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{u}^{(1)} & \boldsymbol{u}^{(2)} & \boldsymbol{u}^{(3)} & \cdots & \boldsymbol{u}^{(n)} \\ | & | & | & & | \end{bmatrix} \in \boldsymbol{R}^{n \times n}$$

$$z^{(i)} = (\boldsymbol{U}_{reduce})^T \, x^{(i)}$$

$$z^{(i)} = \underbrace{\begin{bmatrix} | & | & & | \\ \boldsymbol{u}^{(1)} & \boldsymbol{u}^{(2)} & \cdots & \boldsymbol{u}^{(k)} \\ | & | & & | \end{bmatrix}}_{\boldsymbol{n} \times \boldsymbol{k} \ (\boldsymbol{U}_{reduce})}^T x^{(i)} = \underbrace{\underbrace{\begin{bmatrix} - & (\boldsymbol{u}^{(1)})^T & - \\ - & (\boldsymbol{u}^{(2)})^T & - \\ & \vdots & \\ - & (\boldsymbol{u}^{(k)})^T & - \end{bmatrix}}_{\boldsymbol{k} \times \boldsymbol{n}} \underbrace{x^{(i)}}_{\boldsymbol{n} \times \boldsymbol{1}}}_{\boldsymbol{k} \times \boldsymbol{1}}$$

PCA Algorithm

- After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

$$\textbf{Sigma} = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}\right)\left(x^{(i)}\right)^{T}$$

```
[U,S,V] = svd(sigma);
```

```
Ureduce = U(:,1:k);
```

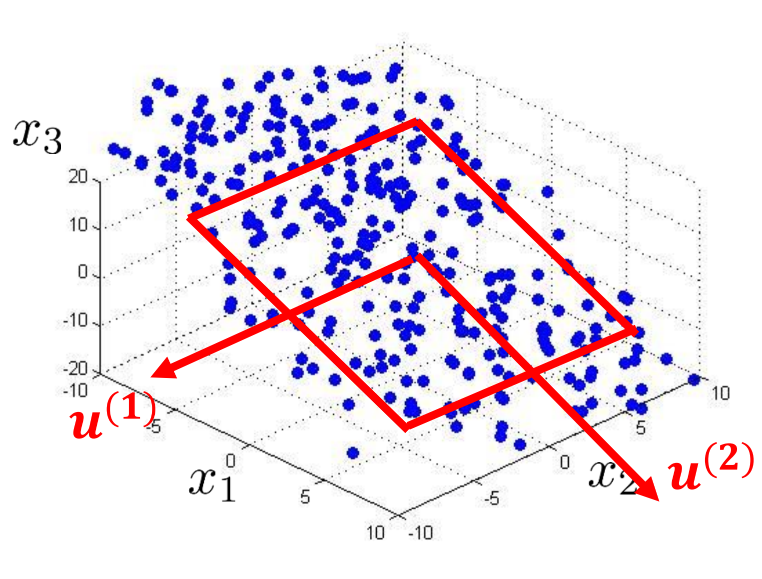
```
Z=Ureduce'*x;
```

PCA Algorithm

- PCA algorithm
  - Preprocessing (mean normalization & feature scaling)
  - Calculate sigma (covariance matrix)
  - Calculate eigenvectors with **svd**
  - Take k vectors from U (**U$_{reduce}$= U(:,1:k);**)
  - Calculate z (**z =U$_{reduce}$' * x;**)

PCA Algorithm

- Reduce data from 3D to 2D.
  - $x^{(i)} \in R^3$ ➔ $z^{(i)} \in R^2$



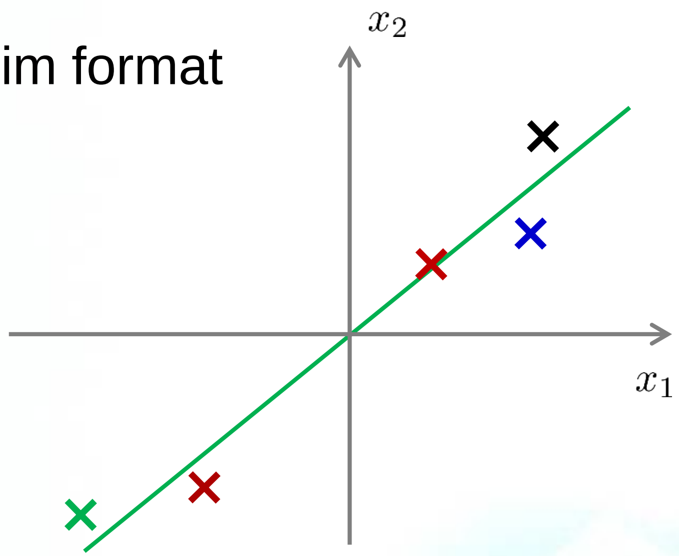
Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Reconstruction from Compressed Rep.

- PCA: A compression algorithm

- **Decompression of** the data
  - from lower dim back to a higher dim format



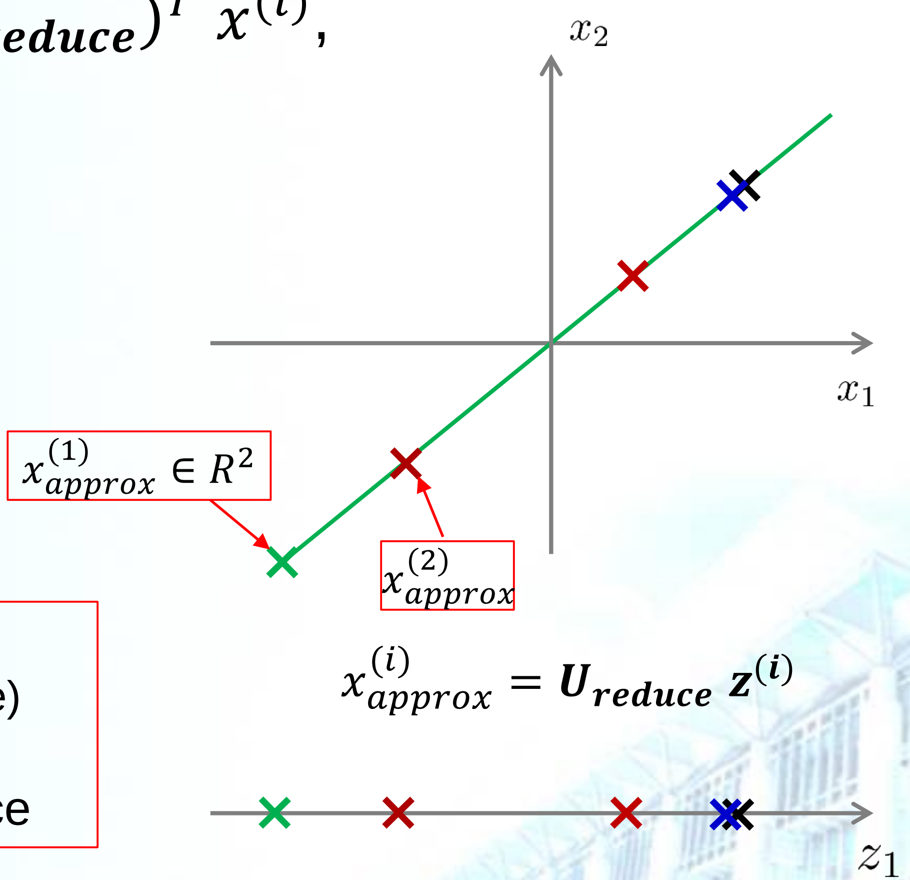
$$z = U_{reduce}^{T} x$$

Reconstruction from Compressed Rep.

- Reconstruction
  - Given a point z^1, how can we go back to the 2D space?

- Considering $z^{(i)} = (\boldsymbol{U_{reduce}})^T \, x^{(i)}$,

$$x_{approx}^{(i)} = \boldsymbol{U_{reduce}} \, \boldsymbol{z}^{(i)}$$



- We lose some of the information
  - (i.e. everything is now ON that line)
- but it is now projected into 2D space

Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Choosing k (# of Principal Components)

- PCA tries to minimize average squared projection error

$$\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2$$

- Total variation in data

$$\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2$$

- Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2}\leq 0.01 \qquad (1\%)$$

  - “99% of variance is retained”

Choosing k (# of Principal Components)

- Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - x_{approx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01$$

  - “99% of variance is retained”
  - If this ratio is small, then the numerator is small
    - The numerator is small when $x^{(i)}$ is very close to $x_{approx}^{(i)}$
      - i.e. we lose very little information in the dimensionality reduction, so when we decompress, we regenerate the same data
  - Often can significantly reduce data dimensionality while retaining the variance

Choosing k (# of Principal Components)

- Algorithm:
  - Try PCA with $k = 1$
  - Compute
    - $U_{reduce}, \ z^{(1)}, \dots, z^{(m)},$
    - $x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$
  - Check if
    - $\dfrac{\frac{1}{m}\sum_{i=1}^{m}\left\| x^{(i)} - x_{approx}^{(i)} \right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\| x^{(i)} \right\|^2} \le 0.01$

Choosing k (# of Principal Components)

- Algorithm:
  - Try PCA with $k = 1$
  - Compute
    - U_{reduce}, $z^{(1)}, \ldots, z^{(m)}$,
    - $x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$
  - Check if
    - $\boxed{\dfrac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x^{(i)}_{approx}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2}} \le 0.01$

- `[U,S,V] = svd(Sigma)`

  $k = 3$

  - $S = \begin{bmatrix} s_{11} & & & & \mathbf{0} \\ & s_{22} & & & \\ & & s_{33} & & \\ & & & \ddots & \\ \mathbf{0} & & & & s_{nn} \end{bmatrix}$

  - For given k

    $$\boxed{1 - \frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}}} \le 0.01$$

    $$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \ge 0.99$$

Choosing k (# of Principal Components)

- `[U,S,V] = svd(Sigma)`

- Pick the smallest value of k for which

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \geq 0.99$$

(99% of variance retained)

Outline

- Data compression
- Data visualization
- Principal component analysis problem formulation
- Principal component analysis algorithm
- Reconstruction from compressed representation
- Choosing the number of principal components
- Advice for applying PCA

Supervised Learning Speedup

- Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$x^{(i)} \in R^{10000}$ ← 100 × 100 image

- Extract inputs
  - Unlabeled dataset: $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in R^{10000}$

$$\downarrow \text{PCA}$$

$$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in R^{100}$$

- New training set: $(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$

- Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set.
  - This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

  - How choose k?
    - % of variance retained

- Visualization
  - Typically $k = 2$ or 3
    - We can plot these values ($k = 2$ or 3)

Application of PCA

- A bad use of PCA
  - Use it to prevent overfitting
  - Use $z^{(i)} \in R^k$ instead of $x^{(i)} \in R^n$ to reduce the number of features $k < n$ (e.g. $k = 1{,}000, n = 10{,}000$)

➔Thus, fewer features, less likely to overfit.

➔ This might work OK, but is not a good way to address ovefitting

- Use regularization for preventing overfitting
  - $\min_{\theta} \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$

Application of PCA

- A bad use of PCA
  - Use it to prevent overfitting

➔

- PCA does NOT use the labels $y^{(i)}$
  - PCA is just looking our inputs $x^{(i)}$ to find a lower dimensional approximation to our data
  - PCA throws away some information (or reduce the dim of our data) without knowing what the values of y is
    - This might be probably OK if we are keeping most of the variance (e.g. 99% of the variance)
      - But it might also throw away some valuable information

Application of PCA

- A bad use of PCA
  - Use it to prevent overfitting

➔

- Retaining 99% of the variance or 95% or whatever,
  - Just using regularization will often give us AT LEAST as good a way for preventing overfitting
  - Regularization knows what the values of y are,

    so is less likely to throw away some valuable information
    - while PCA does not make use of the labels and

      it is more likely to throw away valuable information

Application of PCA

- PCA is used for compression or visualization
  - good

Application of PCA

- But PCA is sometimes used where it should not be
  - Design of ML system with PCA form the outset
    - Get training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$
    - Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
    - Train logistic regression on $(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$
    - Test on test set:
      - Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$.
      - Run $h_\theta(z)$ on $\left(z_{test}^{(1)}, y_{test}^{(1)}\right), \left(z_{test}^{(2)}, y_{test}^{(2)}\right), \dots, \left(z_{test}^{(2)}, y_{test}^{(2)}\right)$

Application of PCA

- But PCA is sometimes used where it should not be
  - Design of ML system with PCA form the outset
    - Get training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$
    - Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
    - Train logistic regression on $(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$
    - Test on test set:
      - Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$.
      - Run $h_\theta(z)$ on $\left(z_{test}^{(1)}, y_{test}^{(1)}\right), \left(z_{test}^{(2)}, y_{test}^{(2)}\right), \ldots, \left(z_{test}^{(2)}, y_{test}^{(2)}\right)$

- How about doing the whole thing without using PCA?
  - Before implementing PCA, first try running whatever we want to do with the original/raw data $x^{(i)}$.
  - Only if that does not do what we want, then implement PCA and consider using $z^{(i)}$.

References

- https://www.coursera.org/learn/machine-learning
- http://www.holehouse.org/mlclass/14_Dimensionality_Reduction.html